```
/*
TinyTrackGPS.ino - A simple track GPS to SD card logger.
TinyTrackGPS v0.4

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or
  modify
  it under the terms of the GNU General Public License as published
  by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/
  licenses/>.
*/

/*
     Programa de localizacion por gps que graba las posiciones en
     un fichero de texto cada segundo, de forma diaria.

     Conectar módulo SD con pin CS (naranja) en pin 10 arduino.

     Conectar módulo NMEA-6M (gps) pines 8,9 (9 - pin rx negro)

     Conectar LCD 16x2 pines 2,3,4,5,6,7 (2-amarillo , 3-azul,
     4-rojo, 5-azul oscuro, 6-verde, 7-blanco)
*/
// Include libraries.
#include <Ticker.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <SD.h>
#include <LowPower.h>
#include "UTMconversion.h"

```

```
46    File GPSFile;
47    char GPSLogFile[] = "YYYYMMDD.csv"; // Formato de nombre de
      fichero. YYYY-Año, MM-Mes, DD-Día.
48    boolean SDReady;
49    float flat, flon;
50    unsigned long age;
51
52    /* Código de demostración uso de librería TinyGPS.
53       Requiere uso de librería SoftwareSerial, se presupone que
      disponemos
54       de un dispositivo GPS serie de 9600-bauds conectado en pines
      9(rx) y 8(tx).
55    */
56    TinyGPS gps;
57    GPS_UTM utm;
58    #define UTM_LCD
59    #define PIN_SELECT 8
60    boolean pin = LOW;
61    SoftwareSerial gps_serial(9, 8);
62    int year_actual;
63    byte month_actual, day_actual;
64    byte hour_prev, minute_prev, second_prev;
65
66    void GPSData();
67    Ticker GPSRefresh(GPSData, 1000);
68
69    /* Constantes para declaracion del LCD */
70    const int LCD_NB_ROWS = 2;
71    const int LCD_NB_COLUMNS = 16;
72    /* Crea el objeto lcd */
73    LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
74
75    // DEFINICION DE CARACTERES PERSONALIZADOS
76    byte alt[8] = {
77      0b00000100,
78      0b00001110,
79      0b00011111,
80      0b00000100,
81      0b00000100,
82      0b00000100,
83      0b00000100,
84      0b00000100,
85    };
86
87    byte ant[8] = {
88      0b00001110,
89      0b00010001,
```

```
    0b00010101,
    0b00010001,
    0b00000100,
    0b00000100,
    0b00001110,
    0b00000000,
  };

  byte sd[8] = {
    0b00001110,
    0b00010001,
    0b00011111,
    0b00000000,
    0b00000000,
    0b00010111,
    0b00010101,
    0b00011101,
  };


  void setup(void) {
    boolean config = 0;
    Serial.begin(9600);
    gps_serial.begin(9600);

    pinMode(10, OUTPUT);

    Serial.print(F("Initializing SD card..."));

    SDReady = SD.begin(10);
    (SDReady) ? Serial.println(F("Done.")) :
    Serial.println(F("FAILED!"));

    /* Declaramos pin para selector visor coordenadas */
    pinMode(PIN_SELECT,INPUT_PULLUP);

    /* Iniciaización del LCD */
    lcd.begin(LCD_NB_COLUMNS, LCD_NB_ROWS);

    lcd.createChar(0, alt);
    lcd.createChar(1, ant);
    lcd.createChar(2, sd);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Waiting for"));
    lcd.setCursor(0, 1);
    lcd.print(F("GPS signal."));
```

```
137
138     Serial.print(F("Waiting for GPS signal..."));
139
140     do {
141       while (gps_serial.available())
142       {
143         if (gps.encode(gps_serial.read())) // Comprueba que ha
          recibido una sentencia del GPS.
144           {
145             int year;
146             byte month, day, hour, minute, second, hundredths;
147             unsigned long age;
148
149             gps.crack_datetime(&year, &month, &day, &hour, &minute,
              &second, &hundredths, &age);
150             if (sprintf(GPSLogFile, "%04d%02d%02d.csv", year, month,
              day) > 0) {
151               config = true;
152               Serial.println(F("Done."));
153               }
154             year_actual = year;
155             month_actual = month;
156             day_actual = day;
157             hour_prev = hour;
158             minute_prev = minute;
159             second_prev = second;
160
161             // Si no existe el fichero lo crea y añade las cabeceras.
162             if (SDReady && !SD.exists(GPSLogFile)) {
163                if (GPSFile = SD.open(GPSLogFile, FILE_WRITE)) {
164                  Serial.print(F("New GPSLogFile, adding heads..."));
165                  GPSFile.println(F("Time,latitude,longitude,alt,utm"))
                  ;
166                  Serial.println(F("Done."));
167                  GPSFile.close();
168                } else {
169                  Serial.println(F("** Error creating GPSLogFile.
                  **"));
170                }
171             }
172           }
173       }
174     }while(!config);
175
176     if (SDReady) {
177       Serial.print(F("Filename: "));
178       Serial.println(GPSLogFile);
```

```
    }
    Serial.println(F("Configuration ended."));
    lcd.clear();
    GPSRefresh.start();
  }

  void loop(void) {
  //  boolean gps_ok = false;

  //  do {
    while (gps_serial.available())
      gps.encode(gps_serial.read()); //{
  //        gps_ok = true;
  //    }
  //  }while(!gps_ok);

    GPSRefresh.update();
  }

  void GPSData() {
    int year;
    byte month, day, hour, minute, second, hundredths;
    unsigned long age;
    int elevation;
    float f_elevation;
    char timestr[]= "00:00:00";
    char utmstr[] = "30S 123456 1234567";
    long timeidle = millis();

    gps.f_get_position(&flat, &flon, &age);
    f_elevation = gps.f_altitude();
    elevation = abs((int)f_elevation);
    utm.UTM(flat, flon);
    sprintf(utmstr, "%02d%c %ld %ld", utm.zone(), utm.band(),
    utm.X(), utm.Y());
    if (pin != digitalRead(PIN_SELECT)) {
      lcd.clear();
      pin = digitalRead(PIN_SELECT);
      }
    if (pin == LOW) {
      lcd.setCursor(0, 0);
      if (utm.zone()<10) lcd.print("0");
      lcd.print(utm.zone());
      lcd.print(utm.band());
      lcd.setCursor(4, 0);     //lcd.print(F(" "));
      lcd.print(utm.X());
      lcd.setCursor(11, 0);   //lcd.print(F(" "));
      lcd.write((byte)1);
```

```
      //lcd.setCursor(13, 0);   //lcd.print(F(" "));
      if (gps.satellites() < 10) lcd.print(F("0"));
      lcd.print(gps.satellites());
      lcd.setCursor(15, 0);
      SDReady ? lcd.write((byte)2) : lcd.print(F("-"));

      // New line
      lcd.setCursor(1, 1);   // lcd.setCursor(0, 1); lcd.print(F("
      "));
      lcd.print(utm.Y());
      lcd.setCursor(9, 1);   //lcd.print(F(" "));
      lcd.write((byte)0);
      if (elevation >= 1000) lcd.setCursor(10, 1) ;
        else if (elevation >= 1000) lcd.print(F(" "));
        else if (elevation >= 100) lcd.print(F("  "));
        else if (elevation >= 10) lcd.print(F("   "));
        else lcd.print(F("    "));
      lcd.print(elevation);
      lcd.print(F("m"));
      }
    else {
      lcd.setCursor(0, 0);
      lcd.print(F("LAT="));
      lcd.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
      lcd.setCursor(0, 1);
      lcd.print(F("LON="));
      lcd.print(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
      };

    gps.crack_datetime(&year, &month, &day, &hour, &minute, &second,
    &hundredths, &age);
    if (age != TinyGPS::GPS_INVALID_AGE){
      sprintf(timestr, "%02d:%02d:%02d,", hour, minute, second);
      Serial.print(timestr);
    }

    if (year != year_actual || month != month_actual || day !=
    day_actual) {
      sprintf(GPSLogFile, "%04d%02d%02d.csv", year, month, day);
      year_actual = year;
      month_actual = month;
      day_actual = day;

      // Si no existe el fichero lo crea y añade las cabeceras.
      if (SDReady && !SD.exists(GPSLogFile)) {
        if (GPSFile = SD.open(GPSLogFile, FILE_WRITE)) {
          Serial.print(F("New GPSLogFile, adding heads..."));
```

```
270           GPSFile.println(F("Time,latitude,longitude,alt,utm"));
271           Serial.println(F("Done."));
272           GPSFile.close();
273           }
274           else {
275           Serial.println(F("** Error creating GPSLogFile. **"));
276           }
277       }
278     }
279     if (!((hour_prev == hour) && (minute_prev == minute) &&
      •     (second_prev == second))) {
280     if (SDReady && (GPSFile = SD.open(GPSLogFile, FILE_WRITE))) {
281       Serial.print(F("Open GPSLogFile to write..."));
282       GPSFile.print(timestr);
283       GPSFile.print(flat,6);
284       GPSFile.print(",");
285       GPSFile.print(flon,6);
286       GPSFile.print(",");
287       GPSFile.print(elevation);
288       GPSFile.print(",");
289       GPSFile.println(utmstr);
290       GPSFile.close();
291       Serial.println(F("Done."));
292       hour_prev = hour;
293       minute_prev = minute;
294       second_prev = second;
295     } else if (!GPSFile){
296       Serial.println(F("** Error opening GPSLogFile. **"));
297     }
298   } else Serial.println(F("** GPS signal lost. **"));
299   timeidle -= millis();
300   timeidle += 1000L;
301   Serial.println(timeidle);
302   if (timeidle > 500L)
303     LowPower.idle(SLEEP_500MS, ADC_OFF, TIMER2_ON, TIMER1_ON,
      •     TIMER0_ON, SPI_ON, USART0_ON, TWI_OFF);
304   else if(timeidle > 250L)
305     LowPower.idle(SLEEP_250MS, ADC_OFF, TIMER2_ON, TIMER1_ON,
      •     TIMER0_ON, SPI_ON, USART0_ON, TWI_OFF);
306   else if(timeidle > 120L)
307     LowPower.idle(SLEEP_120MS, ADC_OFF, TIMER2_ON, TIMER1_ON,
      •     TIMER0_ON, SPI_ON, USART0_ON, TWI_OFF);
308   else if(timeidle > 60L)
309     LowPower.idle(SLEEP_60MS, ADC_OFF, TIMER2_ON, TIMER1_ON,
      •     TIMER0_ON, SPI_ON, USART0_ON, TWI_OFF);
310   }
311
```